Set up a git repository and access with https

In this blog post I show you how you can set up a git repository and access with https on a CentOS 7 System. The most tutorials I could find used the ssh protocol to access the repository. Not exactly what I wanted because I don't want to create new linux system users for all the colleagues who only need access to the repository and to have only one git user for all the users together seems not to be a professional solution.
Fortunately a simple CGI program to serve the contents of a Git repository to Git clients accessing the repository over http:// and https:// protocols exists. The module is called git-http-backend. In this tutorial I'll show you, how you can set up your own git repository server and access it with https and a self signed certificate from your eclipse installation. I'm sure you'll save a lot of time if you read this blog post because there exists a lot of stumbling blocks and SELinux needs to be convinced that git-http-backend can work. As web server we use Apache2 (httpd) because git-http-backend is a CGI script. Theoretically Nginx can also handle CGI Scripts but you have to install an additional application server for your task, such as uWSGI for example. You can find more information about this topic here. To use Nginx instead of Apache2 makes it a lot more complicated. Believe me I tried by myself.

Install Git

```
# yum install git
```

After the installation you should find git-http-backend on your system, verify with:

```
  # ls -al /usr/libexec/git-core/git-http-backend
```

Install Apache2

```
  # yum install httpd
```

And make sure httpd starts on boot time:

```
  # systemctl enable httpd
```

After the installation we need an additional apache module called mod_ssl. mod_ssl is needed for https access. Just install the module with:

```
  # yum install mod_ssl
```

Next, verify the module installation. On redhat systems the apache configuration is by default held in one file /etc/httpd/conf/httpd.conf. All modules are loaded from this file, and can be disabled by commenting out the appropriate LoadModule statement. By default, all modules are loaded when they have beeen installed. If you don't want to load a module, then edit the appropriate file in /etc/httpd/conf.modules.d and comment the load line with #.

Create virtual host

In the next step we create the virtual host my.domain.ch, replace my.domain.ch with your host for example git.test.ch and make sure git.test.ch points to your server where you have installed the apache daemon. Create the file /etc/httpd/conf.d/secure.my.domain.ch.conf with following content:

```
          SSLEngine on        SSLCertificateFile /data/git/ssl/git.cer
t.crt      SSLCertificateKeyFile /data/git/ssl/git.cert.key      SetEn
vIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown        Serv
erName my.domain.ch      ErrorLog /var/log/httpd/git-error.log      Cu
stomLog /var/log/httpd/git-access.log combined        # GIT Config
  SetEnv GIT_PROJECT_ROOT /data/git/repositories      SetEnv GIT_HTTP_
EXPORT_ALL        # Route Git-Http-Backend      ScriptAlias / /usr/lib
exec/git-core/git-http-backend/        # Require Acces for all resourc
es               AuthType Basic         AuthName "Private"
Require valid-user          AuthUserFile /data/my.htpasswd
```

Create a git repository

Note the repository location within the configuration file above:

```
  SetEnv GIT_PROJECT_ROOT /data/git/repositories
```

Now we create the necessary directory and initialise a first git repository:

```
  # mkdir -p /data/git/repositories  # cd /data/git/repositories  # gi
t --bare init my-
project.git  # chown -R apache:apache /data/git/repositories
```

Create Certificate

Note the ssl certificate and ssl certificate key file location within the configuration file above:

```
  SSLCertificateFile /data/git/ssl/git.cert.crt  SSLCertificateKeyFile
 /data/git/ssl/git.cert.key
```

Next we create the needed files, answer the question for "Common Name" with your real host, which you want to user for accessing the repository, e.g: my.domain.ch:

```
  # mkdir /data/git/ssl  # cd /data/git/ssl  # openssl req -new > git.
cert.csr  # openssl rsa -in privkey.pem -out git.cert.key  # openssl x
509 -in git.cert.csr -out git.cert.crt  -req -signkey git.cert.key -da
ys 3650
```

Create the password file

Notice the AuthUserFile within the configuration file above, this is the location where apache checks users and passwords for basic authentication create the file my.htpasswd as follows and provide a password when asked:

```
  # yum provides \*bin/htpasswd  # yum install httpd-tools-2.4.6-18.el
7.centos.x86_64  # htpasswd -c /data/my.htpasswd myuser
```

SELinux configurations

Try to restart apache now with:

```
  # systemctl restart httpd.service
```

As you can see apache isn't starting anymore, because SELinux is blocking the httpd daemon. A message like this occurs:

```
  Job for httpd.service failed. See 'systemctl status httpd.service' a
nd 'journalctl -xn' for details.
```

Type the following for more information:

```
  # systemctl status httpd.service  httpd.service - The Apache HTTP Se
rver     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabl
ed)     Active: failed (Result: exit-code) since Di 2014-12-30 16:27:0
0 CET; 9s ago    Process: 12132 ExecStop=/bin/kill -WINCH ${MAINPID} (
code=exited, status=0/SUCCESS)    Process: 12130 ExecStart=/usr/sbin/h
ttpd $OPTIONS -DFOREGROUND (code=exited, status=1/FAILURE)   Main PID:
 12130 (code=exited, status=1/FAILURE)    Dez 30 16:26:59 ??? httpd[12
130]: AH00526: Syntax error on line 5 of /etc/httpd/conf.d/secure.my.d
omain.ch.conf:  Dez 30 16:26:59 ??? httpd[12130]: SSLCertificateFile:
file '/data/git/ssl/git.cert.crt does not exist or is empty  Dez 30 16
:26:59 ??? systemd[1]: httpd.service: main process exited, code=exited
, status=1/FAILURE  Dez 30 16:27:00 ??? systemd[1]: Failed to start Th
e Apache HTTP Server.  Dez 30 16:27:00 ??? systemd[1]: Unit httpd.serv
ice entered failed state.
```

Also check the files /var/log/messages, /var/log/httpd/error_log and /var/log/audit/audit.log for additional information. Use the following command to generate human readable reports from audit.log.

```
  # yum whatprovides sealert  # yum install setroubleshoot-server-3.2.
17-2.el7.x86_64  # sealert -a /var/log/audit/audit.log > /path/to/mylo
gfile.txt
```

The error message above tells us /data/git/ssl/git.cert.crt does not exist or is empty. But both of them isn't true. Also the file has read rights. Why apache throws this message? Yes the answer is, the SELinux security file context is wrong. If SELinux blocks an action, this is reported to the underlying application as a normal "access denied" type error to the application. Many applications, however, do not test all return codes on system calls and may return no message explaining the issue or may return in a misleading fashion. In fact the httpd daemon is not allowed to read the file /data/git/ssl/git.cert.crt. To solve this problem we can take over the security context from another file which already has the correct context:

```
  chcon --reference=/etc/pki/tls/certs/ca-bundle.crt /data/git/ssl/git
.cert.crt  chcon --reference=/etc/pki/tls/certs/ca-bundle.crt /data/gi
t/ssl/git.cert.key  chcon --reference=/etc/pki/tls/certs/ca-
bundle.crt /data/git/ssl/git.cert.csr
```

Or copy the files to the common locations and adapt your /etc/httpd/conf.d/secure.my.domain.ch.conf accordingly.

```
  # cp /data/git/ssl/git.cert.crt /etc/pki/tls/certs/  # cp /data/git/
ssl/git.cert.key /etc/pki/tls/private/  # cp /data/git/ssl/git.cert.cs
r /etc/pki/tls/private/
```

If you copy the files to the common locations, then the security context will be set automatically. With the following command you can see the security context of a file:

```
  # ls -alZ /data/git/ssl/git.cert.crt  -rw-r--r--. root root unconfin
ed_u:object_r:cert_t:s0  /data/git/ssl/git.cert.crt
```

Make sure only root can read the certificate files:

```
  # chmod 600 /data/git/ssl/git.*
```

After you have set the security context to cert_t the httpd daemon should be able to read the file. Try to restart the deamon now.

```
  # systemctl restart httpd.service
```

No you should see the same problem with the AuthUserFile /data/my.htpasswd which is used for basic authentication. Use the following command to set the correct security context:

```
  # chcon -t httpd_sys_content_t /data/my.htpasswd
```

Restart httpd again, httpd should now start without any problems.

Install a certificate

For a first test we use the command curl on the system where you have installed the git repository to verify the access to our repository, simply use:

```
  curl -u myuser:'mypassword' https://my.domain.ch/my-project.git
```

Because we use a self signed certificate you should face the following error message:

```
  curl: (60) Peer's certificate issuer has been marked as not trusted
by the user.  More details here: http://curl.haxx.se/docs/sslcerts.htm
l    curl performs SSL certificate verification by default, using a "b
undle"   of Certificate Authority (CA) public keys (CA certs). If the
default   bundle file isn't adequate, you can specify an alternate fil
e   using the --cacert option.  If this HTTPS server uses a certificat
e signed by a CA represented in   the bundle, the certificate verifica
tion probably failed due to a   problem with the certificate (it might
 be expired, or the name might   not match the domain name in the URL)
.  If you'd like to turn off curl's verification of the certificate, u
se   the -k (or --insecure) option.
```

As the message tells you, you could turn off the certificate verification and probably you could do the same for your git client afterwards. Nevertheless this is not secure because the client will accept the certificate without verification. Your client sofware can't be sure to talk with the permitted and desired endpoint server. The secure way is to install the certificate on the client, you can do that with:

```
  $ update-ca-trust enable  $ cp /data/git/ssl/git.cert.crt /etc/pki/c
a-trust/source/anchors/  $ update-ca-
trust extract  $ systemctl reload httpd.service
```

Now try again:

```
  # curl -u myuser:'mypassword' https://my.domain.ch/my-project.git
```

If everything works as expected, curl should return without a message nor an error message.

Access with a git client

Now we try to access our repository with a git client, provide the password when asked:

```
  # git clone https://myuser@my.domain.ch/my-project.git
```

One more time you will see an error message saying something like:

```
  fatal: repository 'https://myuser@my.domain.ch/my-
project.git/' not found
```

If you check the error log /var/log/httpd/git-error.log you can see:

```
  [cgi:error] [pid 12890] [client ???.???.???.???:?????] AH01215: Not
a git repository: '/data/git/repositories/my-project.git'
```

Here again SELinux blocks, the process /usr/libexec/git-core/git is not allowed to access /data/git/repositories. You can find more information in the SELinux log file:

```
  # tail -f /var/log/audit/audit.log | grep 'type=AVC'  type=AVC msg=a
udit(1420040376.300:54570): avc:  denied  { create } for  pid=983 comm
="git" name="39" scontext=system_u:system_r:httpd_t:s0 tcontext=system
_u:object_r:file_t:s0 tclass=dir  type=SYSCALL msg=audit(1420040376.30
0:54570): arch=c000003e syscall=83 success=no exit=-13 a0=7af120 a1=1f
f a2=7463656a626f2f2e a3=7fff1fb07c30 items=0 ppid=981 pid=983 auid=42
94967295 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=
48 tty=(none) ses=4294967295 comm="git" exe="/usr/libexec/git-
core/git" subj=system_u:system_r:httpd_t:s0 key=(null)
```

To solve this problem change the security context of /data/git/repositories recursively and permanent with:

```
  semanage fcontext -a -t httpd_git_rw_content_t "/data/git/repositori
es(/.*)?"
```

Access with egit client in Eclipse

Do the following steps to clone your remote git repository.

- Open the git configuration: Window -> Preferences -> Team -> Git -> Configuration and press the button Add Entry...
- Add a new key http.sslVerify with value false
- open the git perspective in Eclipse Window -> Open Perspective -> Other... -> Git
- Press the Icon "Clone a Git Repository and add the clone to this view"
- In the opening Wizard choose Clone URI and press next.
- In the window "Source Git Repository" enter your repository uri for example: https://my.domain.ch/my-project.git
- As Protocol choose https
- Enter your user and password
- Then press next until you reach the window "Local Destination"
- Provide the local path in the field Directory, this is the path where you local git copy will be placed
- Check the box "Import all existing projects after clone finishes" and press Finish

Did you notice the insecure step above? Yes the key http.sslVerify with value false is bad. If you access your git repository like this then you are not protected against Man-In-The-Middle Attacks. The better way is to import the certificate git.cert.crt from the server to your local keystore. Do the following steps to make it more secure.

- Open the git configuration: Window -> Preferences -> Team -> Git -> Configuration and remove the key http.sslVerify
- Try to make a push to your remote repository: Right mouse click on your eclipse project -> Team -> Remote -> Push...
- Provide the repository uri for instance https://my.domain.ch/my-project.git and your user password combination and press next.
- Now you will see an error message of course, because the certificate can't be verified. The error message looks like:

  ```
  Transport Error: Cannot get remote repository refs.  https://git.
  dropbit.ch/course-management.git: cannot open git-upload-pack
  ```

- Copy the file git.cert.crt from the remote repository server to your local host where eclipse is running
- Add git.cert.crt to your local keystore with the keytool command (you should find keytool in the java jdk bin directory):

  ```
    keytool -keystore C:\Java\jdk1.7.0_71\jre\lib\security\cacerts
  -storepass changeit -import -alias git -trustcacerts -v -file c:\
  tmp\git.cert.crt
  ```

For your information, if you did something wrong you can delete a certificate from the keystore with:

```
  keytool -delete -noprompt -alias git -keystore C:\Java\jdk1.7.0
_71\jre\lib\security\cacerts -storepass changeit
```

- Close Eclipse and append the following to your eclipse.ini file:

```
  -Djavax.net.ssl.trustStore=C:\develop\Java\jdk1.7.0_71\jre\lib\
security\cacerts  -Djavax.net.ssl.trustStorePassword=changeit
```

- Restart Eclipse and try again to push something to your remote git repository

The error message above should disappear now. I hope everything is working in your environment. If you have any questions or improvements, don't hesitate to contact me. Thank you for reading my blog post.